Un nouvel algorithme de génération des itemsets fermés fréquents

Huaiguo Fu

CRIL-CNRS FRE2499, Université d'Artois - IUT de Lens
Rue de l'université SP 16, 62307 Lens cedex. France.
E-mail: fu@cril.univ-artois.fr

Motivation

- Develop data mining tools based on formal concept analysis
  - ☞ Concept lattices is an effective tool for data analysis
- Develop scalable concept lattice algorithm to reduce expensive cost when mining very large data
  - ☞ Problem: very large data still bothers us
  - ☞ Situation: applications of concept lattice suffer from the complexity of the concept lattice algorithms for large dataset
  - ☞ One solution: divide-and-conquer

Outline

Concept lattice

- The core of formal concept analysis
- Derived from mathematical order theory and lattice theory
- Study of the generation and relation of formal concepts:
  - how objects can be hierarchically grouped together according to their common attributes
  - generate knowledge rules from concept lattice

Data → Concept lattice → Knowledge

An effective tool

Concept lattice is an effective tool for

- Data analysis
- Knowledge discovery and data mining
- Information retrieval
- Software engineering
- ...

Concept lattice for Data mining (DM)

- Association rules mining (ARM)
  - Frequent Closed Itemset (FCI) mining
    - CLOSE, A-close, Titanic ...
    - CLOSET, CHARM, CLOSET+ ...
- Supervised classification
  - GRAND, LEGAL, GALOIS
  - RULEARNER, CIBLe, CLNN&CLNB ...
- Clustering analysis based on concept lattices
- Data visualization ...

Concept lattice algorithms

- Many concept lattice algorithms
  - Chein (1969)
  - Norris (1978)
  - Ganter (NextClosure) (1984)
  - Bordat (1986)
  - Godin (1991, 1995)
  - Dowling (1993)
  - Kuznetsov (1993)
  - Carpineto and Romano (1993, 1996)
  - Nourine (1999)
  - Lindig (2000)
  - Valtchev (2002) ...
- However, large data impedes or limits the application of concept lattice algorithms
- We need to develop scalable lattice-based algorithm for data mining

Formal context

Definition

Formal context is defined by a triple (O, A, R), where O and A are two sets, and R is a relation between O and A. The elements of O are called objects or transactions, while the elements of A are called items or attributes.

Example

Formal context **D**: (O, A, R)
$O = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
1	×	×					×	
2	×	×					×	×
3	×	×	×				×	×
4	×		×				×	×
5	×	×		×		×		
6	×	×	×	×		×		
7	×		×	×	×			
8	×		×	×		×		

Formal concept

Definition

Two **closure operators** are defined as $O_1 \rightarrow O_1''$ for set O and $A_1 \rightarrow A_1''$ for set A.

$$O_1' := \{a \in A \mid oRa \text{ for all } o \in O_1\}$$

$$A_1' := \{o \in O \mid oRa \text{ for all } a \in A_1\}$$

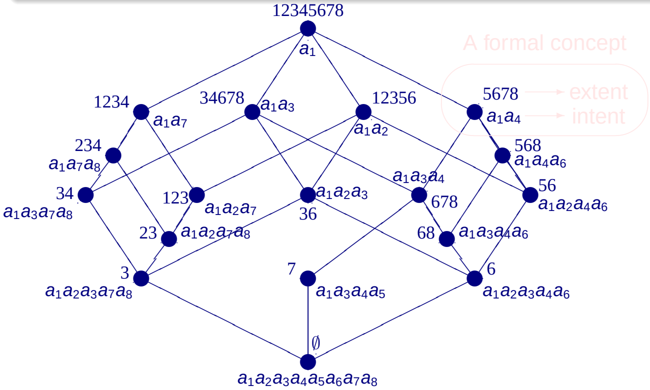
Definition

A **formal concept** of (O, A, R) is a pair (O_1, A_1) with $O_1 \subseteq O$, $A_1 \subseteq A, O_1 = A_1'$ and $A_1 = O_1'$. O_1 is called extent, A_1 is called intent.

Concept lattice

Definition

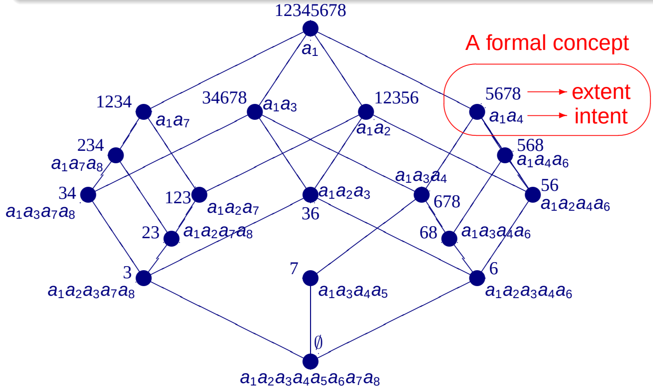
We say that there is a hierarchical order between two formal concepts (O_1, A_1) and (O_2, A_2), if $O_1 \subseteq O_2$ (or $A_2 \subseteq A_1$).
All formal concepts with the hierarchical order of concepts form a complete lattice called **concept lattice**.



Concept lattice

Definition

We say that there is a hierarchical order between two formal concepts (O_1, A_1) and (O_2, A_2), if $O_1 \subseteq O_2$ (or $A_2 \subseteq A_1$).
All formal concepts with the hierarchical order of concepts form a complete lattice called **concept lattice**.



Association rule mining

- Finding frequent patterns, associations, correlations, or causal structures among itemsets or objects in databases
- Two sub-problems
  - mining frequent itemsets
  - generating association rules from frequent itemsets
- Basic Concepts
  - Formal context (O, A, R), an itemset A_1
  - support: $support(A_1) = |A_1'|$
  - frequent itemset: if $support(A_1) \geq minsupport$, given a minimum support threshold *minsupport*

Algorithms of mining frequent itemsets

- Classic algorithm: Apriori and OCD [Agrawal et al., 96]
- Improvement of Apriori
  - AprioriTid,
  - AprioriHybrid
  - DHP (Direct Hashing and Pruning)
  - Partition
  - Sampling
  - DIC (Dynamic Itemset Counting)...
- Other strategies
  - FP-Growth ...
- Problem: too many frequent itemsets if the minsupport is low
  - Solutions:
    - Maximal frequent itemset mining
    - Frequent closed itemset mining

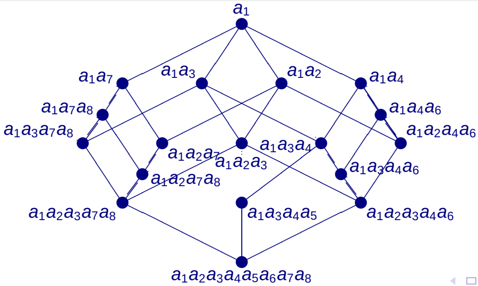
Closed itemset lattice

Definition

For formal context (O, A, R), an itemset $A_1 \subseteq A$ is a **closed itemset** iff $A_1'' = A_1$.

Definition

We say that there is a hierarchical order between closed itemsets A_1 and A_2, if $A_1 \subseteq A_2$. All closed itemsets with the hierarchical order form of a complete lattice called **closed itemset lattice**.

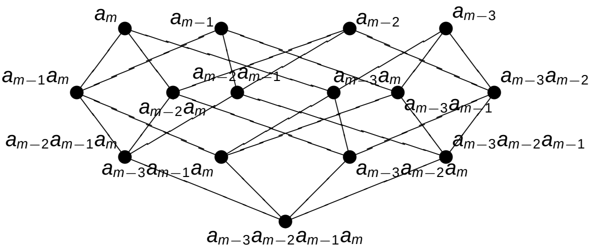


FCI mining algorithms

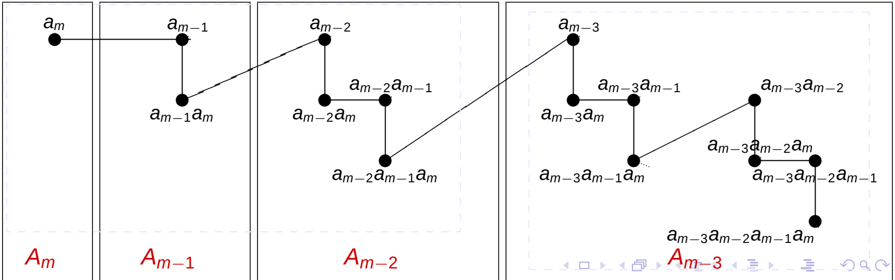
- Apriori-like algorithms
  - Close
  - A-Close
  - ...
- Hybrid algorithms
  - CHARM
  - CLOSET
  - CLOSET+
  - FP-Close
  - DCI-Closed
  - LCM
  - ...

Analysis of search space

- Search space of a formal context with 4 attributes

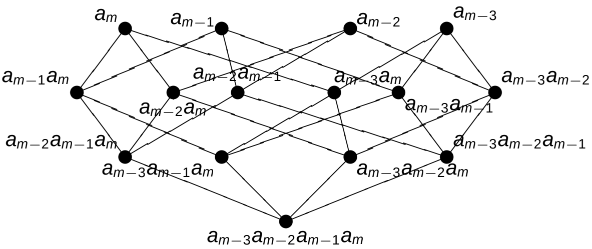


- Search space decomposition

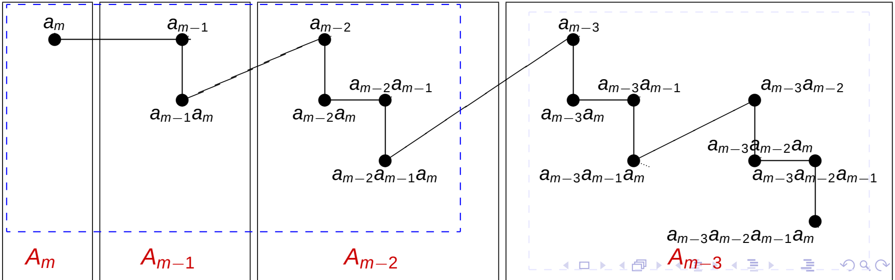


Analysis of search space

- Search space of a formal context with 4 attributes

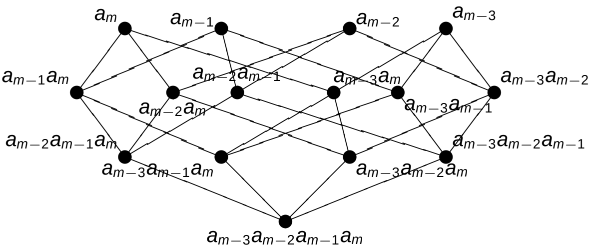


- Search space decomposition

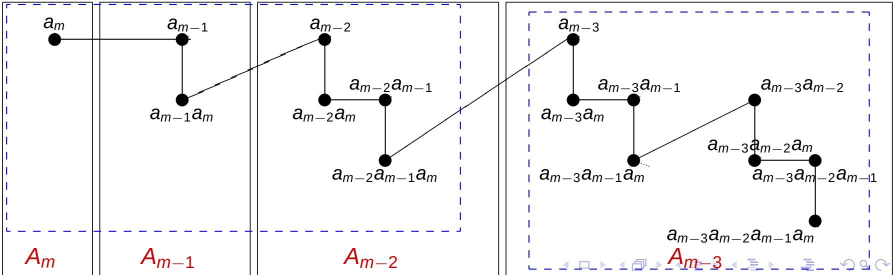


Analysis of search space

- Search space of a formal context with 4 attributes



- Search space decomposition



Analysis of search space

Definition

Given an attribute $a_i \in A$ of the context (O, A, R), a set E, $a_i \notin E$. We define $a_i \otimes E = \{\{a_i\} \cup X \text{ for all } X \subseteq E\}$.

$$\begin{aligned} A_k &= \{a_k\} \cup \left(\bigcup_{\forall X_i \in \cup A_j} \{\{a_k\} \cup X_i\} \right) \\ &= a_k \otimes \{a_{k+1}, a_{k+2}, \cdots, a_m\} \quad k+1 \leq j \leq m \end{aligned}$$

- Problems:
  - ☞ how to balance the size of partitions
  - ☞ whether each partition contains closed itemsets

A strategy of decomposition

- Partitions (for the preceding example: formal context **D**)

$partition_1$	$partition_2$	$partition_3$	$partition_4$
A_8	A_4	A_2	A_1
A_7	A_3		
A_6			
A_5			

- Results: not good!
  - ☞ $Nb_c(partition_1)$=0
    $Nb_c(partition_2)$=0
    $Nb_c(partition_3)$=0
    $Nb_c(partition_1)$=17
- A solution: order the attributes
  - ☞ $Nb_c(partition_1)$=6
    $Nb_c(partition_2)$=6
    $Nb_c(partition_3)$=4
    $Nb_c(partition_1)$=1

Ordered data context

Definition

A formal context is called **ordered data context** if we order the items of formal context by number of objects of each item from the smallest to the biggest one, and the items with the same objects are merged as one item.

Example

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1	×	×					×		×
2	×	×					×	×	×
3	×	×	×				×	×	×
4	×		×				×	×	
5	×	×		×		×			×
6	×	×	×	×		×			×
7	×		×	×	×				
8	×		×	×		×			

Formal context

$\Longrightarrow$

$\Longrightarrow$

	a_5	a_8	a_6	a_7	a_4	a_3	a_2	a_1
1				×			×	×
2		×		×			×	×
3		×		×		×	×	×
4		×		×		×		×
5			×		×		×	×
6			×		×	×	×	×
7	×				×	×		×
8			×		×	×		×

Ordered data context

Folding search sub-space

Definition

An item a_i of a formal context (O, A, R), all subsets of $\{a_i, a_{i+1}, \ldots, a_{m-1}, a_m\}$ that include a_i, form a search sub-space (for closed itemset) that is called **folding search sub-space (F3S)** of a_i, denoted $F3S_i$.

✍ Remark

The search space of closed itemsets:

$F3S_m \cup F3S_{m-1} \cup F3S_{m-2} \cup F3S_{m-3} \cup \cdots \cup F3S_i \cdots \cup F3S_1 \cup \emptyset$

An example

step 1: $(O, A, R) \Longrightarrow (O, A^{\lhd}, R)$, Ordered data context: $a_5\ a_8\ a_6\ a_7\ a_4\ a_3\ a_2\ a_1$

step 2: $DP = 0.5 \Longrightarrow p_1 = 8,\ p_2 = 4,\ p_3 = 2,\ p_4 = 1 \Longrightarrow p_1 \rightarrow a_1,\ p_2 \rightarrow a_7,\ p_3 \rightarrow a_8,\ p_4 \rightarrow a_5$

1	2	3	4	5	6	7	8
a_5	a_8	a_6	a_7	a_4	a_3	a_2	a_1

$$\underbrace{\underbrace{\underbrace{\underbrace{a_5}_{p_4=1}\ a_8}_{p_3=2}\ a_6\ a_7}_{p_2=4}\ a_4\ a_3\ a_2\ a_1}_{p_1=8}$$

Partitions:

choose: $\boxed{a_4\ a_3\ a_2\ a_1}$

remain: $a_5\ a_8\ a_6\ a_7$

choose: $\boxed{a_6\ a_7}$

remain: $a_5\ a_8$

choose: $\boxed{a_8}$

remain: a_5

choose: $\boxed{a_5}$

remain: $\emptyset$

$partition_1 = [a_1, a_7)$: $a_1 \longrightarrow a_4$

$partition_2 = [a_7, a_8)$: $a_7 \longrightarrow a_6$

$partition_3 = [a_8, a_5)$: a_8

$partition_4 = [a_5)$: a_5

step 3: Search the closed itemsets in each partition separately:

$[a_1, a_7)$: $a_1,\ a_2a_1,\ a_3a_1,\ a_3a_2a_1,\ a_4a_1,\ a_4a_3a_1$

$[a_7, a_8)$: $a_7a_1,\ a_7a_2a_1,\ a_6a_4a_1,\ a_6a_4a_2a_1,\ a_6a_4a_3a_1,\ a_6a_4a_3a_2a_1$

$[a_8, a_5)$: $a_8a_7a_1,\ a_8a_7a_2a_1,\ a_8a_7a_3a_1,\ a_8a_7a_3a_2a_1$

$[a_5)$: $a_5a_4a_3a_1$

An example

step 1: $(O, A, R) \Longrightarrow (O, A^{\triangleleft}, R)$, Ordered data context: $a_5\ a_8\ a_6\ a_7\ a_4\ a_3\ a_2\ a_1$

step 2: $DP = 0.5 \Longrightarrow p_1 = 8,\ p_2 = 4,\ p_3 = 2,\ p_4 = 1 \Longrightarrow p_1 \to a_1,\ p_2 \to a_7,\ p_3 \to a_8,\ p_4 \to a_5$

$$\underbrace{\underbrace{\underbrace{\underbrace{\overset{1}{a_5}}_{p_4=1}\ \overset{2}{a_8}}_{p_3=2}\ \overset{3}{a_6}\ \overset{4}{a_7}}_{p_2=4}\ \boxed{\overset{5}{a_4}\ \overset{6}{a_3}\ \overset{7}{a_2}\ \overset{8}{a_1}}}_{p_1=8}$$

Partitions:

choose: $\boxed{a_4\ a_3\ a_2\ a_1}$

remain: $a_5\ a_8\ a_6\ a_7$

choose: $\boxed{a_6\ a_7}$

remain: $a_5\ a_8$

choose: $\boxed{a_8}$

remain: a_5

choose: $\boxed{a_5}$

remain: $\emptyset$

$partition_1 = [a_1, a_7)$: $a_1 \longrightarrow a_4$

$partition_2 = [a_7, a_8)$: $a_7 \longrightarrow a_6$

$partition_3 = [a_8, a_5)$: a_8

$partition_4 = [a_5)$: a_5

step 3: Search the closed itemsets in each partition separately:

$[a_1, a_7)$: $a_1,\ a_2a_1,\ a_3a_1,\ a_3a_2a_1,\ a_4a_1,\ a_4a_3a_1$

$[a_7, a_8)$: $a_7a_1,\ a_7a_2a_1,\ a_6a_4a_1,\ a_6a_4a_2a_1,\ a_6a_4a_3a_1,\ a_6a_4a_3a_2a_1$

$[a_8, a_5)$: $a_8a_7a_1,\ a_8a_7a_2a_1,\ a_8a_7a_3a_1,\ a_8a_7a_3a_2a_1$

$[a_5)$: $a_5a_4a_3a_1$

An example

step 1: $(O, A, R) \Longrightarrow (O, A^{\lhd}, R)$, Ordered data context: $a_5\ a_8\ a_6\ a_7\ a_4\ a_3\ a_2\ a_1$

step 2: $DP = 0.5 \Longrightarrow p_1 = 8,\ p_2 = 4,\ p_3 = 2,\ p_4 = 1 \Longrightarrow p_1 \to a_1,\ p_2 \to a_7,\ p_3 \to a_8,\ p_4 \to a_5$

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ a_5 & a_8 & a_6 & a_7 & a_4 & a_3 & a_2 & a_1 \end{array}$$

$p_4 = 1$

$p_3 = 2$

$p_2 = 4$

$p_1 = 8$

Partitions:

choose: $\boxed{a_4\ a_3\ a_2\ a_1}$

remain: $a_5\ a_8\ a_6\ a_7$

choose: $\boxed{a_6\ a_7}$

remain: $a_5\ a_8$

choose: $\boxed{a_8}$

remain: a_5

choose: $\boxed{a_5}$

remain: $\emptyset$

$partition_1 = [a_1, a_7)$: $a_1 \longrightarrow a_4$

$partition_2 = [a_7, a_8)$: $a_7 \longrightarrow a_6$

$partition_3 = [a_8, a_5)$: a_8

$partition_4 = [a_5)$: a_5

step 3: Search the closed itemsets in each partition separately:

$[a_1, a_7)$: $a_1,\ a_2a_1,\ a_3a_1,\ a_3a_2a_1,\ a_4a_1,\ a_4a_3a_1$

$[a_7, a_8)$: $a_7a_1,\ a_7a_2a_1,\ a_6a_4a_1,\ a_6a_4a_2a_1,\ a_6a_4a_3a_1,\ a_6a_4a_3a_2a_1$

$[a_8, a_5)$: $a_8a_7a_1,\ a_8a_7a_2a_1,\ a_8a_7a_3a_1,\ a_8a_7a_3a_2a_1$

$[a_5)$: $a_5a_4a_3a_1$

An example

step 1: $(O, A, R) \Longrightarrow (O, A^{\lhd}, R)$, Ordered data context: $a_5\ a_8\ a_6\ a_7\ a_4\ a_3\ a_2\ a_1$

step 2: $DP = 0.5 \Longrightarrow p_1 = 8,\ p_2 = 4,\ p_3 = 2,\ p_4 = 1 \Longrightarrow p_1 \to a_1,\ p_2 \to a_7,\ p_3 \to a_8,\ p_4 \to a_5$

$$\underbrace{\underbrace{\underbrace{\underbrace{\overset{1}{a_5}}_{p_4=1}\ \overset{2}{a_8}}_{p_3=2}\ \overset{3}{a_6}\ \overset{4}{a_7}}_{p_2=4}\ \overset{5}{a_4}\ \overset{6}{a_3}\ \overset{7}{a_2}\ \overset{8}{a_1}}_{p_1=8}$$

Partitions:

choose: $\boxed{a_4\ a_3\ a_2\ a_1}$ → $partition_1 = [a_1, a_7)$: $a_1 \longrightarrow a_4$

remain: $a_5\ a_8\ a_6\ a_7$

choose: $\boxed{a_6\ a_7}$ → $partition_2 = [a_7, a_8)$: $a_7 \longrightarrow a_6$

remain: $a_5\ a_8$

choose: $\boxed{a_8}$ → $partition_3 = [a_8, a_5)$: a_8

remain: a_5

choose: $\boxed{a_5}$ → $partition_4 = [a_5)$: a_5

remain: $\emptyset$

step 3: Search the closed itemsets in each partition separately:

$[a_1, a_7)$: $a_1,\ a_2a_1,\ a_3a_1,\ a_3a_2a_1,\ a_4a_1,\ a_4a_3a_1$

$[a_7, a_8)$: $a_7a_1,\ a_7a_2a_1,\ a_6a_4a_1,\ a_6a_4a_2a_1,\ a_6a_4a_3a_1,\ a_6a_4a_3a_2a_1$

$[a_8, a_5)$: $a_8a_7a_1,\ a_8a_7a_2a_1,\ a_8a_7a_3a_1,\ a_8a_7a_3a_2a_1$

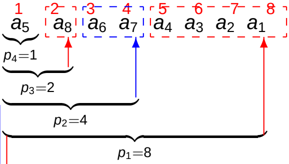
$[a_5)$: $a_5a_4a_3a_1$

An example

step 1: $(O, A, R) \Longrightarrow (O, A^{\triangleleft}, R)$, Ordered data context: $a_5\ a_8\ a_6\ a_7\ a_4\ a_3\ a_2\ a_1$

step 2:

$DP = 0.5 \Longrightarrow p_1 = 8,\ p_2 = 4,\ p_3 = 2,\ p_4 = 1 \Longrightarrow p_1 \rightarrow a_1,\ p_2 \rightarrow a_7,\ p_3 \rightarrow a_8,\ p_4 \rightarrow a_5$



Partitions:

choose: $\boxed{a_4\ a_3\ a_2\ a_1}$

remain: $a_5\ a_8\ a_6\ a_7$

choose: $\boxed{a_6\ a_7}$

remain: $a_5\ a_8$

choose: $\boxed{a_8}$

remain: a_5

choose: $\boxed{a_5}$

remain: $\emptyset$

$partition_1 = [a_1, a_7)$: $a_1 \longrightarrow a_4$

$partition_2 = [a_7, a_8)$: $a_7 \longrightarrow a_6$

$partition_3 = [a_8, a_5)$: a_8

$partition_4 = [a_5)$: a_5

step 3:

Search the closed itemsets in each partition separately:

$[a_1, a_7)$: a_1, a_2a_1, a_3a_1, $a_3a_2a_1$, a_4a_1, $a_4a_3a_1$

$[a_7, a_8)$: a_7a_1, $a_7a_2a_1$, $a_6a_4a_1$, $a_6a_4a_2a_1$, $a_6a_4a_3a_1$, $a_6a_4a_3a_2a_1$

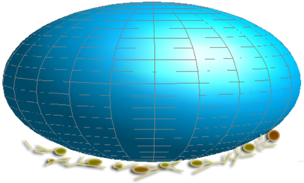
$[a_8, a_5)$: $a_8a_7a_1$, $a_8a_7a_2a_1$, $a_8a_7a_3a_1$, $a_8a_7a_3a_2a_1$

$[a_5)$: $a_5a_4a_3a_1$

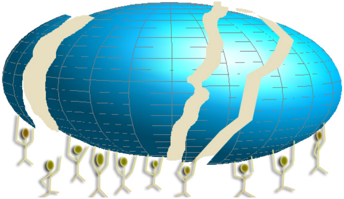
The principle of PFC

PFC algorithm has two steps:

1) Determining the partitions
2) Generating independently all frequent closed itemsets of each partition



$\Rightarrow$



Failure

Success

divide-and-conquer

Determining the partitions

- $(O, A, R) \Rightarrow (O, A^{\lhd}, R)$ where
  $A^{\lhd} = \{a_1^{\lhd}, a_2^{\lhd} \ldots, a_i^{\lhd}, \ldots, a_m^{\lhd}\}$
- Some items of $A^{\lhd}$ are chosen to form an order set P
  - ☞ $P = \{a_{P_T}^{\lhd}, a_{P_{T-1}}^{\lhd} \ldots, a_{P_k}^{\lhd}, \ldots, a_{P_1}^{\lhd}\}$, $|P| = T$, $a_{P_k}^{\lhd} \in A^{\lhd}$
  - ☞ $a_{P_T}^{\lhd} < a_{P_{T-1}}^{\lhd} < \ldots < a_{P_k}^{\lhd} < \ldots < a_{P_2}^{\lhd} < a_{P_1}^{\lhd} = a_m^{\lhd}$
- DP is used to choose $a_{P_k}^{\lhd}$ $(0 < DP < 1)$
  - ☞ $DP = \frac{|\{a_1^{\lhd}, \cdots, a_{P_k}^{\lhd}\}|}{|\{a_1^{\lhd}, \cdots, a_{P_{k-1}}^{\lhd}\}|}$
- The partitions: $[a_{P_k}^{\lhd}, a_{P_{k+1}}^{\lhd})$ and $[a_{P_T}^{\lhd})$
  - ☞ Interval $[a_{P_k}, a_{P_{k+1}})$ is the search space from item a_{P_k} to $a_{P_{k+1}}$
  - ☞

$$\left[a_{P_k}^{\lhd}, a_{P_{k+1}}^{\lhd}\right) = \bigcup_{P_k \le i < P_{k+1} \le P_T} (F3S_i)$$

$$\left[a_{P_T}^{\lhd}\right) = F3S_{P_T}$$

Generating FCIs in each partition

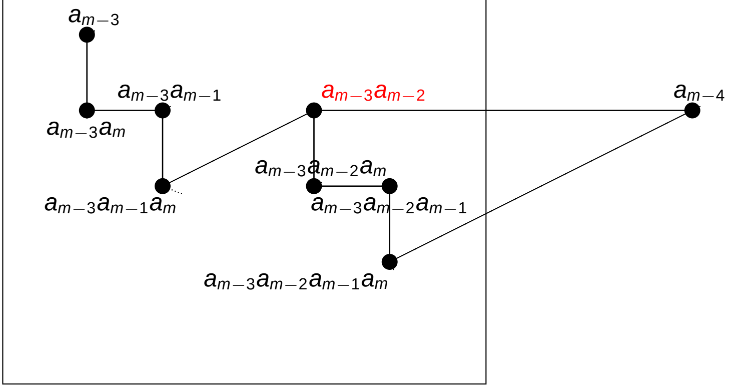
Definition

Given an itemset $A_1 \subset A$, $A_1 = \{b_1, b_2, \ldots, b_i, \ldots, b_k\}$, $b_i \in A$. A_1 is an infrequent itemset. **The candidate of next closed itemset** of A_1, noted $C_{A_1}^{\bowtie}$, is $A_1 \uplus a_i = (A_1 \cap (a_1, a_2, \ldots, a_{i-1}) \cup \{a_i\})''$, where $a_i < b_k$ and $a_i \notin A_1$, a_i is the biggest one of A with $A_1 < A_1 \uplus a_i$ following the order: $a_1 < \ldots < a_i < \ldots < a_m$.

- Partitions are independent and non-overlapped
- For each partition, from an itemset A_1
  - ☞ If $|A_1'| \geq minsupport$, we search the next closure of A_1
  - ☞ If $|A_1'| < minsupport$, we search $C_{A_1}^{\bowtie}$. The closed itemsets between A_1 and $C_{A_1}^{\bowtie}$ are ignored

An example

- Search FCIs with PFC



Theoretical comparison

- Extended comparison of [BenYahia and Mephu, 04]
- Comparison

Algorithm	Format	Technique	Architecture	Parallelism
A-CLOSE	horizontal	Generate-and-test	Sequential	no
CLOSET+	horizontal	Hybrid, FP-tree	Sequential	no
CHARM	vertical	Hybrid, IT-tree, diffset	Sequential	no
PFC	relational	Hybrid, partition	Distributed	yes

Experimentation

- Many FCI algorithms were compared in [FIMI03]
- In order to get a reference of the performance of PFC
- C++ for CLOSET+ vs Java for PFC
- Sequential architecture

Data	**Objects**	**Items**	**Minsupport**	**FCI**	PFC (msec)	CLOSET+	Ref.
audiology	26	110	1	30401	1302	8563	+
soybean-small	47	79	1	3541	516	431	-
lung-cancer	32	228	1	186092	21381	279689	+
promoters	106	228	3	298772	120426	111421	-
soybean-large	307	133	1	806030	357408	364524	+
dermatology	366	130	50	192203	20204	18387	-
breast-cancer-wis	699	110	1	9860	3529	1131	-
kr-vs-kp	3196	75	1100	2770846	1823092	483896	-
agaricus-lepiota	8124	124	100	38347	34815	1462	-
connect-4	67557	126	1000	2447136	1165806	65084	-

- Discussion
  - 3 contexts with good results
    - Density
    - or $|A| >> |O|$

Conclusion

- PFC: a scalable algorithm
  - PFC algorithm can be used to generate FCIs
  - Preliminary performance results show that PFC is suitable for large data
- One limitation of PFC: formal context can be loaded in memory
- Perspectives
  - Analyze context characteristic
  - Optimize the parameter of algorithms
  - Parallel and distributed environment
  - Application